DD2420 Tutorial 2

Magnus Pierrau

January 2020

Task 1

Task 1.1

From the column of `delay(>=2)` in figure 1 we read that students over the age of 23 (last row) have the lowest representation (none) of 2 or more years of delay. The 0 will be an issue since it will propagate through the distribution and might create zero probabilities. In fact, just because we have not observed any one older than 23 without a 2 year delay does not mean that the probability is 0, only that it is relatively low. This is known as the zero-frequency problem.

delay	delay(0)	delay(1)	delay(>=2)	delay(NA)
age(20-23)	0.18627450980392157	0.14285714285714285	0.35294117647058826	0.4375
age(<=20)	0.7696078431372549	0.7142857142857143	0.6470588235294118	0.5
age(>23)	0.04411764705882353	0.14285714285714285	0.0	0.0625

Figure 1: CPD table of `delay` given `age`.

Task 1.2

We use the `ratio` function to get the relative frequency of students with an average computer science grade of $4 < 5$ among those with the same grade in mathematics.

The marginal distributions of the `delay` variable are indeed the same in the fit method as those given by the relative frequency counts. This can be explained by the fact that the MLE for a multinomial distribution, such as that which we are handling when conditioning on the delay variable, is given by precisely the relative frequency counts.

This due to the model being a Naive Bayes model. Since all feature variables are conditionally independent, given the target variable, we can infer each parameter estimate independent of the others.

Task 1.3

We can analogously, by using `avg_mat` instead of `age` produce the table seen in figure 2. By comparing this to the output given by the `ratio` function we conclude that the same result holds for the fitted conditional CPD. This ought to hold for every feature variable, as they all follow independent multinomial distributions when conditioned on the `delay` variable.

delay	delay(0)	delay(1)	delay(>=2)	delay(NA)
avg_mat(2<3)	0.39705882352941174	0.5357142857142857	0.47058823529411764	0.0625
avg_mat(3<4)	0.44607843137254904	0.14285714285714285	0.0	0.125
avg_mat(4<5)	0.11274509803921569	0.0	0.0	0.0625
avg_mat(<2)	0.04411764705882353	0.32142857142857145	0.5294117647058824	0.75

Figure 2: CPD of `delay` given `avg_mat`.

Task 2

Task 2.1

Calling the PGM's `query`-function with arguments `variables=['delay']` and `evidence={'age': '<=20'}` we find from the resulting table that the probability for having zero delay if being in the age group of 20 or lower is 0.8.

In task 1.1 we are asked to evaluate the probability of age given delay, which corresponds to the likelihood of the feature variable given the target variable, whilst in this task we are asked what the probability of delay is, given the age. This can be interpreted as the proportional posterior of the same distribution as in task 1.1.

Task 2.2

By using the same query-command as in Task 2.1, but with `variables=['age']` and `evidence='0'`, we find from the presented CPD-tables (figure 3) that the age group > 23 is the one with the lowest probability of having no delay in their bachelor's degree (0.04). The age group with highest probability is those younger than 20 (0.77).

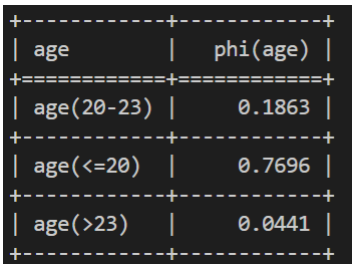

Figure 3: CPD table for `age` conditioned on `delay` being 0.

Task 2.3

By comparing the printouts from the call `ve.query(variables = ['delay'], evidence = 'age': age)` for all values of `age` with those of `ratio(data, lambda t: t['delay']==delay, lambda t: t['age']==age)` for all values of `delay` and `age` we find that they produce the same results.

They should equal since the given PGM structure describes a Naive Bayes model, in which all features are independent, as per previous discussions.

Task 2.4

The `map_query` computes the max-marginal over all variables `age, delay, avg_mat, avg_cs, gender` and returns the outcome that maximises the posterior distribution. The result is shown in figure 4

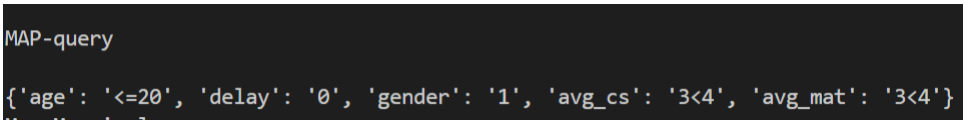

Figure 4: Caption

We note that the most likely outcome for variables `delay` and `age` are 0 and $\leq$ 20, indicating that this is the combination of variable outcomes that maximizes the posterior distribution. This is in line with the previous results from task Task 2.2 showing that the most probable outcome for `age` given `delay = 0` is indeed ≤ 20.

If we give the MAP-query age as variable input and `delay=0` as evidence input, it also produces age¡=20 as output.

Task 3

Task 3.1

The reversed model is created by the `commands model = BayesianModel([('age', 'delay'), ('gender', 'delay'), ('avg_mat', 'delay'), ('avg_cs', 'delay')])` and `model.fit(data)`.

Task 3.2

The CPD for `delay` has $4 \times 3 \times 4 \times 4 \times 2 = 384$ entrires. This corresponds to one entry corresponding to $P(\texttt{delay} = i \mid \texttt{age} = j, \texttt{avg\_cs} = k, \texttt{avg\_mat} = l, \texttt{gender} = m)$ for each i, j, j, l, m. This is because the `delay` response variable is now depending on all four variables.

Task 3.3

The data set is comprised of 265 entries meaning that there are more possible combinations of variable outcomes than there are actual entries. This is seen in the table by noting that there are multiple entries which have a frequency ratio of 0. This will further increase the zero frequency problem.

Task 3.4

By checking for each possible variable combination if there was an instance of that combination in the data we find that there are 321 missing entries out of the total 384 combinations. This means that a fraction of $321/384 = 0.836$ of the possible outcomes where not included in the data set. Given that there are 265 entries, with a maximum of 384 combinations, we could have at most expected to observe $265/384 = 0.31 = 31\%$ of all possible combinations. We observed 16.4% which is within the expected percentage.

In the cases where there were no observations for any outcome of `delay` over all values in *scope*(`delay`) for some given value of the feature variables, then all four probabilities are set equal to 0.25. Since we have no evidence this corresponds to falling back on a uniform prior over the parameter $\theta_{delay,j,k}$.

Task 3.5

We now query the PGM for the marginal distribution over `delay` by `ve.query(variables=['delay'])`. We also compute the relative frequency ratios by `ratio(data, lambda t: t['delay']==delay)` for all instances of the `delay` variable. The results are presented in table 1 below.

We find here that the probabilities are similar, but with some small differences. The absolute errors are of magnitude 10^{-3}, but since the probabilities differ in size for the different outcomes of `delay`, the relative error becomes large for the outcomes with a low relative frequency. The highest relative error

Function	`delay(0)`	`delay(1)`	`delay(≤ 2)`	`delay(NA))`
`ve.query`	0.7611	0.1116	0.0754	0.0520
`ratio`	0.7698	0.1056	0.0642	0.0604
|Err.|	0.0087	0.0060	0.0112	0.0084
Rel. err.	0.0114	0.0538	0.149	0.161

Table 1: Caption

is observed for `delay(NA)`, with a 16.1% relative error, with a the relative error for `delay(≤2)` being in the same magnitude, with 14.9%.

In this case an exact inference should not result in the frequency count, because we are no longer working with a Naive bayes model, and the features are no longer independent given `delay` (they are no longer D-separated).

Task 4

Task 4.1

Thek Kullback-Liebler distance is a form of similiarity measure which measures "how different" a distribution Q is from P according to

$$D_{\mathrm{KL}}(P \parallel Q) = \sum_{x \in \mathcal{X}} P(x) \log \left(\frac{P(x)}{Q(x)} \right). \tag{1}$$

In our case $P(x)$ is the distributions obtained by variable elimination of `model1` and `model2` respectively, and $Q(x)$ of the relative frequency counts (CPD). As the divergence decreases, one can say that Q becomes more simliar to P, and when the divergence is 0 the distributions are equal.

Importantly, the KL-divergence is an asymmetric measure and thus does not qualify as a distance measure. The KL-divergence can also be seen as the difference between the entropy of P and the cross-entropy between P and Q.

Task 4.2

We obtain `inf` values due to the denominator in the KL-divergence being 0 whilst the numerator is non-zero. We also obtain `nan` values when also the numerator is zero. This happens when a query has no support in the data.

Task 4.3

- Row 1: Prints the number of queries with 2 evidence variables given, after sorting out `inf` and `nan`s.
- Row 2: Prints the number of queries with 2 evidence variables given and for which `div(model1)` $<$ `div(model2)` (i.e. for which cases the KL-

divergence is smaller for `model1` than for `model2`, after sorting out `inf` and `nan`s.

- Row 3: Prints the number of queries with 2 evidence variables given and for which `div(model1)` > `div(model2)`, after sorting out `inf` and `nan`s.
- Row 4: Prints the number of queries that had 2 evidence variables given and non-finite divergences for either of the models.
- Row 5: Prints the sum of all KL-divergences between the distribution obtained from PGM `model1` and the CPD.

Task 4.4

N	M1 wins %	M2 wins %	Sum div M1	Sum div M2	# of 'inf/nan'
1	100	0	3.4e-15	1.91	25
2	77.8	22.2	0.427	1.77	65
3	37.5	62.5	0.738	0.55	103
4	0	100	0.164	0.0	110

Table 2: Target: `delay`, 400 data points

As we increase the number of prior conditions we find that Model1 decreases in performance and Model2 increases.

This is due to Model1 using the likelihood while Model2 uses posterior, which includes prior information. Thus, as we introduce more prior information Model2 increases in performance and M1 decreases.

The summed KL-divergence for Model2 is 0 when observing evidence for all variables. This indicates that given enough evidence our fit converges into the MLE (the frequency count). It is also essentially 0 for Model1 when given evidence for just one variable.

Task 4.5

N	M1 wins %	M2 wins %	Sum div M1	Sum div M2	# of 'inf/nan'
1	84.1	15.9	3.30	6.69	42
2	69.7	30.2	4.35	6.74	136
3	66.7	33.3	2.01	3.16	193
4	41.2	58.8	0.45	0.75	240

Table 3: Target: `age`, 1000 data points

Here we note a similar pattern, but not as extreme. The performance of M1 decreases when we introduce more prior information, and increases for M2. However, none of the models score 100% for any N, and none of the divergences

are 0, indicating that none of the inferred distributions are the exact frequency counts.

Task 4.6

N	M1 wins %	M2 wins %	Sum div M1	Sum div M2	# of 'inf/nan'
1	92.7	7.3	2.5	7.6	52
2	71.2	28.8	1.7	4.0	176
3	52	48	1.0	1.5	220
4	52.9	35.3	0.36	0.53	225

Table 4: Targets: `delay` and `age` 1000 data points

We note a similar pattern to that of 3, however, in the end, Model1 still seems to perform better than Model2. The summed KL-divergences are similar as to that of task 4.5.

Task 5

Task 5.1

Since we have a low number of data points we partition the data into a training set and a validation set and perform a k-fold cross validation on it to assess the performance of the models. This way we don't train the models on all data and thus hopefully avoid overfitting. This will make the models more general to new inferences, but at the cost of accuracy on the training set.

Task 5.2

N	Ratio M1 wins, 100% train	Ratio M2 wins, 100% train	Ratio M1 wins, validation	Ratio M2 wins, validation
1	92.7	7.3	66.4	33.6
2	71.2	28.8	65.6	34.4
3	52	48	61.3	38.7
4	52.9	35.3	35.3	40.9

Table 5: Targets:`[age,delay]`, comparison of results between models trained on all data and K-fold cross validation. Pairwise rows don't necessarily sum up to 1 due to cases when KL-divergence were equal. Query size 1000 (325, 80, 25 and 17 after sorting out non-finite values for each N respectively).

Task 5.3

After applying cross validation it becomes clear that the win ratios are much closer than initially presumed. Even though Model1 performs better for $N = 1, 2, 3$, the difference is now considerably smaller.

This indicates that the models were significantly overfitted on the trained data. However, the results still indicate that the naive bayes model has an advantage when not all features are observed as evidence.

Task 6

Task 6.1

The criteria for the K2 score to hold are:

1. The variables in Z are discrete
2. Cases occur independently, given a belief-network model
3. There are no cases that have variables with missing values
4. Before observing the data D, we are indifferent regarding which numerical probabilities to assign to the belief network with structure B_S (no prior beliefs about the network).

Criteria 1,2 and 4 are fulfilled, however, in the dataset there are variables with missing values, with regards to the `NA` values of delay. However, these are not interpreted as NaN's by the code but is rather seen as an discrete outcome of the variable. Therefore the K2-score is applicable, but I am uncertain of what the intention was for this exercise.

Nonetheless, we follow the instructions (using the specified `"NA"` and not `numpy.NaN`) and run the K2 score, defined by

$$\log P(B_S, D) = \log P(B_S) + \sum_{i=1}^{n} \sum_{j=1}^{q_i} \log \frac{(r_i - 1)!}{(N_{ij} + r_i - 1)!} + \sum_{k=1}^{r_i} \log N_{ijk}!. \quad (2)$$

Here r_i is the number of states of $\boldsymbol{X}_i$ (i.e. 2 for `gender`, 4 for `avg_cs`, etc.), $q_i = \prod_{\boldsymbol{X}_j \in Pa(\boldsymbol{X}_i)} r_j$, is the number of possible configurations of the parent set $Pa(\boldsymbol{X}_i)$ of $\boldsymbol{X}_i$, N_{ijk} is the number of instances in the data D where the variable $\boldsymbol{X}_i$ takes its k-th value x_{ik} and the variables in $Pa(\boldsymbol{X}_i)$ (the parents of $\boldsymbol{X}_i$) take their j-th configuration. N_{ij} is the number of instances in the data D where the variables in $Pa(\boldsymbol{X}_i)$ take their j-th configuration, i.e. we marginalise over all $k = 1, ..., r_i$.

This corresponds to setting a Dirichlet prior over all hyperparameters of the multinomial distribution of the model and setting the prior hyperparameters/psuedo-count to 1. The psuedo-count is a way to "overcount" every occation of variable

combination, so that there are no zero-counts occuring in the data. This is a type of additive smoothing.

Running the K2-scores we find that `model1` receives a K2-score of -1037.7 and `model2` a score of -1096. We are aiming to maximize the score (indicating that the model fits data better), and thus `model1` proves to be slightly (albeit marginally with a 5% margin) better.

Task 6.2

Instead of considering which model that gets a lower KL-divergence we find the model which maximizes the posterior, given some various prior assumptions (listed in task 6.1).

Task 6.3

The exhaustive search returns `[('avg_cs', 'avg_mat'), ('avg_cs', 'age'), ('avg_mat', 'delay')]` as the model with the structure which explains the data best. The joint distribution of this PGM model can be factorized as

$$P(\texttt{age}, \texttt{avg\_cs}, \texttt{avg\_mat}, \texttt{delay}, \texttt{gender}) \tag{3}$$

$$= P(\texttt{delay} \mid \texttt{avg\_mat})P(\texttt{avg\_mat} \mid \texttt{avg\_cs})P(\texttt{age} \mid \texttt{avg\_cs})P(\texttt{avg\_cs})P(\texttt{gender}), \tag{4}$$

and its structure is visualized in figure 5.

We note especially that the `gender` variable does not at all affect the queries on any of the other variables, and is effectively completely isolated without edges to the other variables (all variables are independent of it). This means that when we do a query on some variable, the `gender` variable will always be fully marginalized over, summing to 1 and effectively never ipmacting any query.

Task 6.4

In an undirected graph with 4 nodes there are $\frac{4\times 3}{2} = 6$ (four vertices, each with 3 possible edge connections, divided by two for double counting) possible connections to be made between all nodes. In our case, where we have a directed graph, allowed to be cyclical for ease of calculation, this means that each edge can take 3 different formats, either $i \to j$, (implying i influences j), $i \leftarrow j$ (j influences i) or no arrow (no dependence between i and j). This means that there are $3^6 = 729$ possible graph configurations.

Given that an exhaustive search over 4 variables takes 10 seconds, this implies that each graph takes approximately $\frac{10}{729} \approx 0.0137$ seconds to calculate.

If we have 5 variables then there are $\frac{5\times 4}{2} = 10$ possible connections to be made between all nodes. With the same argument as above, this gives

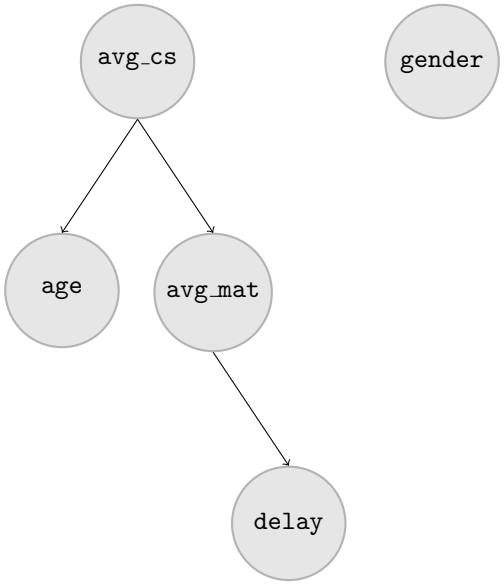

Figure 5: Graphical model of best model according to exhaustive search and HillClimbSearch using K2 as score structure.

$3^{10} = 59049$ possible graph configurations, giving an estimated execution time of $\frac{10}{729} \times 59049 = 810$ s $= 13.5$ min.

The search thus has has a time complexity of $\mathcal{O}(3^{\frac{N(N-1)}{2}}) = \mathcal{O}(3^{N^2})$ which quickly becomes unwieldy as N increases ($N = 6$ would require 54 hours).

Task 6.5

Running a hill climb search using the HillClimbSearch function using K2 as score structure we receive the same model for both four and five variables; `[('avg_cs', 'avg_mat'), ('avg_cs', 'age'), ('avg_mat', 'delay')]` (figure 5). This is the same model as we received from the exhaustive search.

When running the hill climb search for BIC-score as score structure we instead receive the model `[('avg_cs', 'avg_mat'), ('avg_mat', 'delay')]`, visualized in figure 6, for both four and five variables. The underfit manifests itself through the loss of connection between the `age` variable and the other feature variables, giving a loss in specificity and an increase in generality of the model.

Task 6.6

Probabilistic graphical models can be used in order to visualize relations between variables, and make inference on them using special algorithms. It is especially

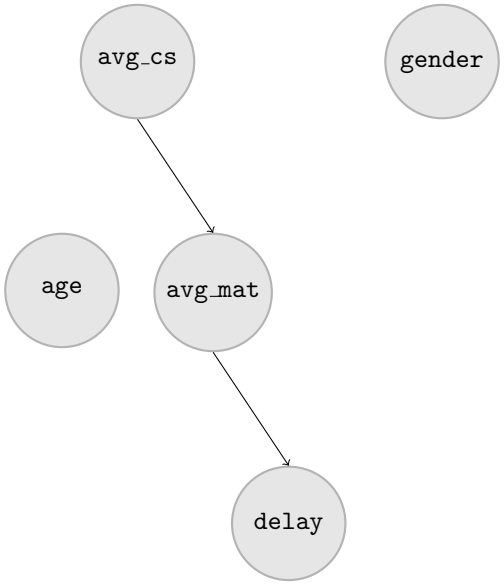

Figure 6: Graphical model of best model according to HillClimbSearch algorithm usin BIC-score as score structure.

useful when there is prior information known about the variables and their inherent relations to each other. However, in this case, we are not certain of how the variables are related, causing us to have to make strong assumptions on their relationships. The Naive Bayes model assumes all feature variables are independent, which is seldom the case in reality, but can nonetheless provide satisfactory approximations at a low computational cost.

Given the low number of data points, and low number of variables, an option to a PGM in this case would have been a decision tree. However, this must be trained anew once new information is acquired, something that the PGM handles much more elegantly.